基于 ARM+FPGA 平台的二值神经网络加速方法研究 *

孙孝辉 [1]，宋庆增 [1]，金光浩 [1+]，姜文超 [2]

(1. 天津工业大学 计算机科学与软件学院，天津 300387; 2. 广东工业大学 计算机学院，广州 510006)

摘　要：目前，现有的卷积神经网络由于其结构复杂且依赖的数据集庞大，难以满足某些实际应用或者计算平台对运算性能的要求和能耗的限制。针对这些应用或计算平台，对基于 ARM+FPGA 平台的二值化算法进行了研究，并设计了二值神经网络，该网络减少了数据对存储单元的需求量，也降低了运算的复杂度。在 ARM+FPGA 平台内部实现时，通过将卷积的乘累加运算转换为 XNOR 逻辑运算和 popcount 等操作，提高了整体的运算效率，降低了对能源和资源的消耗。同时，根据二值神经网络中数据存储的特点，提出了新的行处理改进算法，提高了网络的吞吐量。总之，该实现方式在 GOPS、能源和资源效率方面均优于现有的 FPGA 神经网络加速方法。

关键词：二值神经网络；现场可编程门阵列；异或运算；行处理算法

中图分类号：TP391　　doi: 10.3969/j.issn.1001-3695.2018.08.0614

Research of binary neural network acceleration method based on ARM+FPGA platform

Sun Xioahui[1], Song Qingzeng[1], Jin Guanghao[1+], Jiang Wenchao[2]

(1. *School of Computer Science & Software Engineering*, *Tianjin Polytechnic University*, *Tianjin* 300387, *China*; 2. *School of Computers*, *Guangdong University of Technology*, *Guangzhou* 510006, *China*)

Abstract: At present, the existing convolutional neural network has complicated structure and bases on huge dataset. So it has difficulties in meeting the requirement of computing performance and limitation of energy consumption requested by some practical applications or computing platforms. This papaer studied the binary algorithm based on ARM+ FPGA platform and designed a binary neural network aiming at these applications or platforms. This work reduces the demand for data storage units and simplifies the computational complexity. When implemented in the ARM+ FPGA platform, the convolution multiply-accumulate operation is converted into XNOR logic and popcount operation, which improves the overall operation efficiency and declines the consumption of energy and resources. At the same time, based on the characteristics of data storage in binary neural network, a new row processing algorithm is proposed to improve the throughput of the network. In a word, This implementation is superior to the existing FPGA neural network acceleration methods in terms of GOPS, energy and resource efficiency.

Key words: binary neural network; ARM+FPGA; XNOR; row-processing algorithm

0　引言

卷积神经网络(convolutional neural networks, CNN)是当前最为重要的深度学习算法之一，被广泛应用于计算机视觉[1]、机器翻译[2]、语音识别[3]、人脸检测[4]等领域，都取得了非常好的效果。随着 CNN 及其改进算法的进一步发展，对运算平台的存储容量和计算能力也提出了较高的要求，这就限制了 CNN 等算法在功耗和性能受限的嵌入式平台上的发展。因此，很多研究人员开始提出各种模型压缩算法和技术[5~7]，即在容忍一定的精度下降的代价下，降低 CNN 的运算强度和模型参数的容量。其中，最重要的模型压缩方法之一是用低精度的定点数代替高精度的浮点数，比如 8 bit 定点数代替 32 bit 的浮点数，这种方法能将参数的存储需求降低到原来的四分之一。极端的情况是将 32 bit 浮点数压缩为二值的定点数（1 bit），即所谓的二值神经网络（binary neural network,BNN）[6]。相对于传统 CNN，BNN 这些方法不仅极大地减少了对内存容量的需求，也改变了传统卷积的计算方式，降低了运算的复杂度。

二值神经网络非常适合在 ARM+FPGA 的架构实现。当对参数和特征图进行二值化之后，乘累加操作可以被更简单的异或逻辑(XOR)和按位计数(popcount)所代替，对 FPGA 资源的消耗将大大降低。针对 BNN，可以在 FPGA 上设计特殊的流水结构，进一步加速 BNN 网络的运算速度。与此同时，BNN 网络中并不适合 FPGA 加速的部分可以在 ARM 上运行，保证了实现的灵活性。本文主要研究 BNN 如何在 ARM+FPGA 架构的设计与实现。主要工作如下：针对 Cifar-100 数据集，在 ARM+FPGA 的硬件平台上设计和实现了 BNN 加速器；加速器采用 HLS[8]设计方法，具有更高的灵活性，同时也缩短了开发周期；分别在不同的开发板上进行了测试，并与其他平台在速度、精度和功耗做了对比，实验结果表明本文实现的加速器具备较大的优势。

1　相关工作

基于 FPGA 的卷积神经网络的加速优化措施中，可以通

收稿日期：2018-08-04；**修回日期**：2018-10-15　**基金项目**：国家自然科学基金资助项目（61702366，61602342，61602344，51607122）；广东省科技计划项目（2017B010124001，2017B090901005）；天津市自然科学基金资助项目（16JCYBJC42300，17JCQNJC04500）

作者简介：孙孝辉（1993-），男，河南省商丘人，硕士研究生，主要研究方向为深度学习；宋庆增（1980-），男，天津人，副教授，博士，主要研究方向为深度学习、高性能计算；金光浩（1979-），男（通信作者），吉林延吉人，讲师，博士，主要研究方向为深度学习、高性能计算(jinguanhao@tjpu.edu.cn)；姜文超（1977-），男，山东潍坊人，讲师，博士，主要研究方向为数据挖掘、大数据分析、云计算.

过将网络中的卷积运算转换为运算效率更高的矩阵乘矩阵操作，其中主要的转换算法包括 GEMM 变换[9]和 Winograd 变换[10]，此外还可以通过快速傅里叶变换[11]将时域的卷积转换为频域的乘积来进行优化。除了对网络中的运算进行优化之外，还可以通过阵列的收缩[12]、SIMD[13]、循环优化[14, 15]和设计空间探索[16]等方法来优化网络中数据的传输路径，减小数据之间传输的延迟。

表 1　神经网络模型

Table 1　Neural network model

模型结构	Conv1	Conv2	Pool	Conv3	Conv4	Pool	Conv5	Conv6	Pool	FC1	FC2	FC3
输入特征图	3	128	128	128	256	256	256	512	512	8192	1024	1024
输出特征图	128	128	128	256	256	256	512	512	512	1024	1024	100
特征图维度	32	32	16	16	16	8	8	8	4	1	1	1
BN 参数量	512	512		1024	1024		2048	2048		4096	4096	400
输出所占内存	128K	128K	32K	64K	64K	16K	32K	32K	8192	1024	1024	100
权重所占内存	3456	144K		288K	576K		1.1M	2.3M		8.0M	1.0M	100K

对卷积神经网络加速除了计算变换和数据路径优化外，还可以对网络的权重进行裁剪[17]，增加模型的稀疏性，减少网络的运算次数。同时网络中的卷积核可以通过低秩近似[18]的方法，来减少推断过程中乘法的数量，这些方法通过减少网络的运算量达到了优化的效果。

除了上述的优化方法外，在 FPGA 实现上还可以使用包括定点算法[19]、动态定点算法[20]、二值量化[5, 21, 22]、伪二值量化[23, 24]以及随机计算[25]等近似计算的方法来减少数据的精度来加速网络的运算。这些方法不仅降低了网络中运算的复杂度，还能充分发挥 FPGA 定点计算的潜能。

2　二值神经网络

2.1　网络结构

二值神经网络是一种对 CNN 网络模型进行压缩的实现方法，即将网络中的权重参数和特征图进行了二值化。最初的想法是将权重参数二值化[5]，减少参数的存储量，而后[6]进一步对网络中的特征图也作了压缩处理，在减少存储量的同时也降低了计算的复杂度，并对训练二值神经网络过程中的反向传播问题给出了解决方案。二值神经网络中的二值化是指将权重和激活函数二值化到+1 和-1 的实现。在文献[5]中提出了两种算法，分别是随机性和确定性二值化方法。尽管随机性二值化比确定性二值化更符合实际情况，但在量化时要生成随机位，不利于硬件化实现。因此本文使用了如下的确定性二值化方法：

$$x^b = sign(x) = \begin{cases} +1 & if\ x \ge 0 \\ -1 & otherwise \end{cases} \tag{1}$$

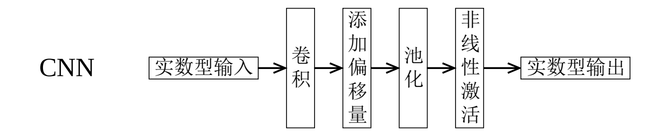
其中: x^b 表示二值化之后的数据（权重/特征图）；x 是原始值。

二值神经网络中为了减少在网络传输过程中由二值化运算带来的分布信息丢失问题，在网络结构中添加了一个新的层次结构，即批归一化(batch normalization, BN)[26]，其运算表达式为

$$y = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}\gamma+\beta \tag{2}$$

其中：x、y 分别表示数据的输入和输出；μ 和 σ2 是每个特征图的平均值和方差值；γ 和 β 是网络训练得到的参数；ϵ 是一个很小的数值保证运算时分母不为零。批归一化层在保证原始数据分布信息稳定性的同时，也加速了网络的训练过程。在网络模型进行推断的过程中，所有的参数都是固定的，所以在实现的过程中，只需关心将式(2)应用到每一个输入的特征图的像素上，而且每个特征图都有唯一对应的批量归一化参数。

BNN 与 CNN 的网络结构定义如图 1 所示。图 1 中 CNN 网络的层次顺序结构为卷积层、偏移层、池化层和激活函数层。其中输入和输出的数据都是实数类型；而 BNN 网络去掉了偏移层，在卷积层之后直接池化，而且为了减少特征图分布信息的丢失，在池化之后添加了 BN 层，除了第一层的输入数据外，每层经过二值激活函数之后输入和输出数据都是二值的。



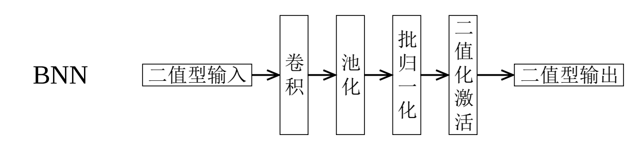


图 1　CNN 与 BNN 的网络结构比较

Fig. 1　Comparison of CNN and BNN

2.2　网络模型

本文设计的二值神经网络模型包括有六个卷积层和三个全连接层。每个卷积层和全连接层之后都有 BN 层和激活层，而池化层是每两个卷积层之后添加一个，也就是在第 2 层、第 4 层、第 6 层卷积之后有一个池化层。卷积层中卷积核的大小为 3×3，池化层中设置采样器的大小为 2×2，步长为 2，对特征图数据取最大值。其中第一层卷积输入的数据为图片的原始数据是浮点类型，这与其余层的数据类型不一样，而每层的权重参数都是二值的。每层的详细参数设置如表 1 所示。其中 Conv 表示卷积层，FC(full connect)表示全连接层。表 1 中根据网络结构分别设置了每层特征图的输入和输出个数，并分别计算了每层的输出和权重所占内存大小，每个数据按照 1 bit 存储来计算。BN 层经过训练之后得到的参数都是浮点类型的数据，经过对卷积之后的数据处理之后得到了中间的浮点型数据，最后经过激活函数将中间的计算结果转换为输出的二值化数据。

2.3　模型训练

本文使用 Theano 和 Lasagne 的深度学习框架[6]，实现表 1 定义的网络模型。模型中使用 cifar-100[27]数据集进行训练和验证，数据集包括有 6 万张 32×32 大小的 3 通道 RGB 图像，其中包括真实世界中的鸟类、猫、飞机、汽车、青蛙等，每个照片的像素值由 0~255 的整数表示。这些图片被分成了 50 000 张训练数据和 10 000 张测试数据，每个数据集都包括 100 个类别，这 100 个类别又被细分成了 20 个大类，每个图像都带有 1 个小类的标签和 1 个大类的标签。

二值神经网络模型中使用了指数衰减学习率，用 ADAM 方法来最小化铰链损失，数据的批大小设置为 50，并使用批归一化来加速训练。实验使用训练集中最后的 5 000 个样本作为验证集，训练了 500 个周期，最后的识别结果如表 2 所示。最终网络中总的权重大小为 13.37 M，其中包括卷积层 4.36 M，全连接层 9.01 M。

3　FPGA 实现

3.1　软硬件划分

基于 ARM+FPGA 架构的设计相较于传统的 FPGA 设计模式，不仅能够缩短开发的周期，而且在设计过程中也可以将性能、集成度和灵活性等因素考虑进来，这不仅降低了成本和功耗，还增强了特性和性能。

本文在针对 ARM+FPGA 架构设计神经网络的前向推断过程时，由于主要的运算阶段在卷积和全连接两部分，为了让平台中各部分发挥出各自最佳的性能，在软硬件划分时，在 FPGA 内部实现主要的网络运算过程，在 ARM 处理器上对输入图片和每层参数进行预加载和分配，并对 FPGA 的初始阶段进行控制。系统的软硬件划分如图 2 所示。硬件逻辑部分主要由计算单元、片内存储器、DMA 和有限状态机控制器组成，系统处理部分主要包括 CPU、内存控制器等。

硬件逻辑部分 DMA 主要用来在运算阶段预加载权重数据到片内存储器中，并将最终的预测结果写回片外存储器。片内存储器主要用来存储计算单元运算之后的结果数据，并将 DMA 读取的数据保存下来。计算单元内部主要是将输入的数据和权重在运算时通过使用一些优化措施，加快网络的运算过程，并输出网络最终的预测结果。

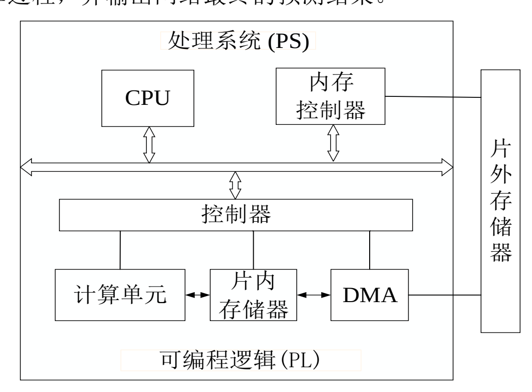


图 2　加速器系统架构

Fig. 2　Architectural diagrams of accelerator

系统处理部分首先将片外存储器中存储的输入特征图和每层的参数数据读入内存之中，按照每层的超参量对参数进行分配，再将分配好的数据和控制信息传输到 FPGA 内部的存储器中。从表 1 可知，特征图和权重分别占据了不同的内存比重，在 FPGA 内部仅仅是保存这些参数，就会消耗很大一部分存储资源。为了最大化地利用存储资源，同时保证系统最佳的性能表现，对 FPGA 器内部计算过程中产生的中间数据的存储部分作了一些改进，详细的介绍将在 4.2 节给出。使用这种方法系统处理部分就可以按照网络层执行的顺序，依次将特征图和参数数据全部或者部分传输到 FPGA 的片内存储器上，保证数据的传输效率。

3.2　硬件设计

根据软硬件划分的结果，针对网络处理的数据类型和运算量的不同，在计算单元内部分成了三个主要的模块来分别处理卷积和全连接，即定点卷积计算模块、二值卷积计算模块和全连接计算模块。每个模块按照网络的层次结构来处理相应的特征图数据，并产生最终的预测结果。

定点卷积计算模块主要用来处理第一层输入的原始图片数据，在该模块内部通过线性缓冲器来接收输入的 3 通道 32x32 大小的图片，然后将对应的像素值转换成 20 位的定点数据。由于输入的权重是二值的，计算过程中使用符号的反转来代替卷积中的乘法。本文方法对这三个通道数据采取完全并行的处理方式，将输入的数据添加到三个缓冲区中，每个循环卷积核在缓冲区中流动，并计算一个 3x3x3 的卷积，其结果通过批归一化和二值化，每循环产生一个输出位。由于第一层卷积占用了很少的运行时间，所以并没有用过多的硬件资源来进行加速。

二值卷积计算模块是计算单元里最重要的组成部分，它用来处理其余的二值卷积层，占据了系统大部分的运行时间。因此该计算模块在处理不同尺寸的输入特征图时，也必须保持高吞吐量和高效的资源利用率。为了在模块内部更有效地进行卷积操作，需要对多行输入数据进行缓冲，以便同时进行访问。但标准的行缓冲器每个周期只能存储单行数据，且只能处理固定缓冲区长度的数据，当输入数据的宽度小于缓冲区的宽度时，就会造成缓冲区利用率不足，导致吞吐量损失。因此，为了能充分利用硬件资源，并解决行缓冲的问题，本文方法设计了可变宽度的行缓冲器，如图 3 所示。可变行缓冲器相较于普通的缓冲器解决了硬件资源利用率不足的问题，并且每个周期可以输出并缓存一行新的特征图数据。在二值卷积计算的过程中，首先从片上存储器读取输入特征图，根据特征图的宽度将数据进行重新组织排序；然后加载到行缓冲器中，卷积模块通过在行缓冲器上滑动产生运算结果，并将它们的累加和输出到缓冲区中，作为下次卷积的输入。

全连接模块执行最后的全连接计算并返回预测的结果，该模块在执行时通过循环展开将特征图数据与权重数据按位执行 XNOR 运算，然后使用 popcount 对结果位求和。与二值计算模块对中间数据处理的方式类似，都是在缓冲区对卷积结果求累加和，并在处理完所有的输入后应用二值化，输出最终的结果。

硬件逻辑部分的片内存储器，由于要同时保存上、下两层之间的输出特征图，本文在设计时根据表 1 中每层的最大参数量，实现了两个内存大小为 128 K 的缓冲器来保存中的运算结果，并对缓存里的数据采取轮转访问的形式，充分利用内存资源，减少片内外数据的传输。

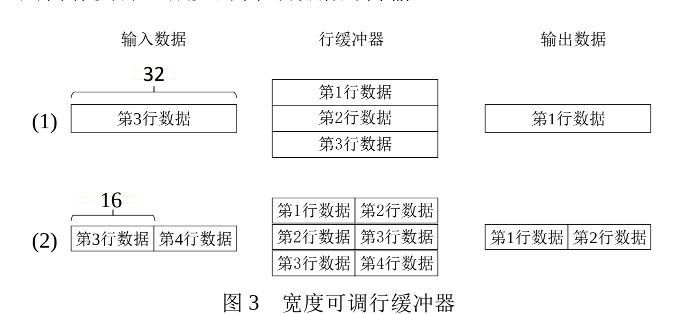


图 3　宽度可调行缓冲器

Fig. 3　Variable-width line buffer

3.3　软硬件接口

本文使用 Xilinx SDSoC 作为 BNN 应用的主要设计工具，在 BNN 加速器的软硬件划分中，软件部分主要来控制整个程序的运行和读取片外存储器里的数据到内存中，在硬件部分主要进行大量的并行计算任务。由于权重和特征图在硬件部分存放和读取的方式不同，在接口实现时采用了折中的方案。

首先，为了减少运算过程中软硬件之间频繁的数据访问，

将系统处理部分的数据通过值传递的方式传输到硬件存储器内部，相较于址传递的访问形式，虽然占用了一些内存空间，但节省了交互的时间开销。

为了保证数据传输的高效性，权重数据需要保存在一段连续的物理内存空间里，并通过 FIFO 流来接收数据。这种数据接收方式更符合权重数据每层依次读取的特性。将对应接口的数据传输方式确定之后，SDSoC 会生成组件单元，组件的两端分别对应 PS 和加速器，中间的控制器就是设定的数据移动方式。

最终通过调用 SDSoC 将软件程序中标记为硬件实现的 C/C++代码部分综合成硬件的 RTL 实现，并通过指定的接口指令生成 PS 与 PL 内存之间的数据传输模块和所需的 DMA 模块。同时在程序中确保产生的中间数据存放在片内存储器上，并且通过连续排列神经网络的数据和权重，保证网络的吞吐量，而片外数据的传输只在第一次和最后一次调用计算单元和加载权重时产生。

4　实验

实验选择在 ZedBoard 和 ZCU102 上评估本文的设计，其中 ZedBoard 代表了一个低成本的 Xilinx Zynq-7000 SoC 包含 7Z020 型号的 FPGA 以及双核 ARM Cortex-A9 嵌入式处理器，ZCU102 作为一个高端的 Zynq UltraScale+ MPSoC 包含 ZU9EG 型号的 FPGA 以及四核 ARM Cortex-A53、双核 ARM Cortex-R5 和图形处理单元的嵌入式处理器。

本文使用 Xilinx SDSoC 2017.4 作为主要的设计工具，利用 Vivado HLS 和 Vivado 将 C/C++的代码，编译成 FPGA 运行的格式。图 2 中的系统结构中的 CPU 分别对应于 ZedBoard 和 ZCU102 器件中的 Cortex-A9 嵌入式处理器和 Cortex-A53 处理器，硬件逻辑中的计算单元主要是基于 FPGA 内部的 LUT 进行实现，并在每个计算单元内部分别调用了不同的可配置逻辑块和输入输出单元，而片内存储器分别对应于 BRAM、FIFO、RAM 等缓冲存储资源。

本文在实现过程中，将计算单元中的三个卷积计算函数分别进行了封装，对应网络整体的推断过程，根据每层所处的运算关系，调用相应的函数，函数处理之后的数据之间的传输是在片内存储器上进行的，存储器上存储着每层运算的特征图结果，整体网络的运算逻辑通过 SDSOC 进行综合实现之后，生成对应的 FPGA 上的硬件逻辑结构。

本文将实现的设计还与两个服务器级的计算平台和一个嵌入式平台进行了比较：一个 Intel Xeon E5-2640 多核处理器 (CPU)，一个 NVIDIA Tesla K80 (GPU)和嵌入式 NVIDIA Jetson TX2 (mGPU)。CPU 和 GPU 的基准线根中提供的代码进行调整，在运算过程中分别调用各自的优化库来进行运算加速。

在运行 BNN 时，器件的功率是通过功率监视器获得的，ZedBoard 和 ZCU102 上的空闲功率分别为 4.4 W、23.4 W，系统运行时的最大功率分别为 4.7 W、23.6 W，这表明 FPGA 在运行过程中的动态功耗还是非常低的。其中吞吐量(giga–opera- tions- per-second, GOPS)计算总的加法与乘运算量，在 BNN 中将每个二进制的异或、翻转和加法分别作为一个运算操作。

4.1　精度分析

二值神经网络模型中的一个复杂因素是二值化数据参与卷积运算时与边缘填充之间的相互作用。二值神经网络将每个激活值二值化为-1 或+1，但每个输入特征图都用零填充边缘，这意味着卷积可以看到最多三个值：-1、0、+1。而 Ternary-NN[24]网络就是考虑了这种情况，将激活函数的值分成了三部分，相较于二值数据，就不再考虑边缘填充带来的影响。而二值神经网络针对三个值在存储时可能就需要两个数据位来保存运算符。本文使用+1 的填充方式来重新训练网络，消除了零点并建立了一个真正的二值化 CNN。这个+1 填充的 BNN 在训练时实现了 58.74%的准确率，在 C/C++的 FPGA 中实现了 57.24%的准确率，仅比原来的稍差。对于 FPGA 实现，使用+1 填充更符合本文的网络设计，考虑到硬件资源的节约，还是采取 0 填充的方式。表 2 比较了对相同的测试数据集 Cifar-100，网络模型的测试结果。

4.2　性能分析

本文的实验将加速器整体的性能与表 3 中的各种基准线进行了比较，由于原始吞吐量在很大程度上受限于设备尺寸，还比较了功耗和每瓦吞吐量。与 mGPU 相比两款 FPGA 的运行时间性能分别提升了 8.和 29.5 倍，每瓦的吞吐量比 mGPU 分别高了 14.1 和 9.4 倍。与 x86 处理器上相比也分别取得了 2 倍和 6.7 倍的加速效果。而与 GPU 相比，两款设计在性能上分别相差了 18.6 和 5.5 倍。但正如预期的一样，它们的功耗要低得多，每瓦的吞吐量也要高很多。其中 Conv1 表示第一层的卷积，Conv2-5 表示的二值卷积层，FC1-3 是最后的全连接层，最后一行显示了各平台每瓦的效率。

表 2　网络模型在 Cifar-100 测试的结果

Table 2　Network model test results in Cifar-100

	模型结构	边缘填充	识别准确度
Ternary-NN[24]	6Conv+2FC+SVM	0	51.6%
BNN (python)	6Conv+2FC+Softmax	0	60%
BNN (python)	-	+1	58.74%
BNN(C++/FPGA)	-	0	57.24%

表 3　计算平台性能比较

Table 3　Performance comparison of computing platforms

	mGPU[6]	CPU[6]	GPU[6]	ZedBoard (this papaer)	ZCU102 (this papaer)
Conv1	-	0.63	0.002	0.088	0.026
Conv2-5	-	13.6	0.38	3.63	1.54
FC1-3	-	0.99	0.015	2.3	0.679
总运行时间	65	14.8	0.397	7.36	2.2
时间加速比	1.0x	4.4x	163.7x	8.8x	29.5x
功耗	7.5	95*	300	4.7	23.6
每瓦吞吐量	2.05	0.71	8.4	28.91	19.26
每瓦吞吐量	2.05	0.71	8.4	28.91	19.26

4.3　功耗分析

由于 BNN 的相对新颖性，目前选定数据集 Cifar-100，设计对应的加速网络，衡量消耗每个资源的吞吐量和每瓦特的吞吐量作为了比较的基准线。

表 4 比较了本文的实现与文献中发现的最前沿的 FPGA 加速器实现方案，表中的对应数字都是从对应的论文中检索得到。其中对比的平台有两个是相同的设备都为 ZedBoard 型号，另一个对比较大容量的 FPGA。本文的 BNN 加速器在吞吐量方面超越了目前最好的 FPGA 加速器，并且也更加节省资源和能源。最终的数据结果也证明 BNN 在算法实现上比 CNN 更适合在 FPGA 上实现，它能更加有效地利用硬件资源。

5　结束语

本文设计了一种基于 ARM+FPGA 的二值神经网络的加

速器，完整地实现了针对 Cifar-100 数据集的网络前向传播过程。BNN 网络对存储需求的减少和二值运算的特性，使其实现方式非常适合 FPGA 的组织结构，通过设计新的加速器架构，使得加速器在单位面积吞吐量和每瓦特的吞吐量方面都优于现有的全精度网络，大幅提升了运算效率。最近低精度的 CNN 的研究也一直在取得进展，并在 ImageNet 的数据集上取得了近乎最好的识别效果[7, 24]。对未来低精度 CNN 的实现，仍需进一步探索缩小模型的算法并设计更大更复杂的加速器，以满足未来大计算量和实时性的需求。

表 4　神经网络 FPGA 加速器比较

Table 4　Comparison of FPGA based neural network accelerators

平台	查找表容量	查找表	时钟	功耗	数据精度	GOPS	GOPS/kLUT	GOPS/Watt
Stratix-V-GSD8[28]	695	120	120	19.1	8-16b	117.8	0.98	6.17
Zynq-7Z045[29]	218.6	182.6	150	9.6	16b	137.0	0.75	14.3
Zynq-7Z020[30]	53.2	43.2	100	-	-	12.73	0.29	7.27
ZedBoard (this papaer)	53.2	38.71	143	4.7	1b	167.7	4.43	35.68
ZCU102 (this papaer)	274.08	132.95	300	23.6	1b	561.0	4.22	23.77

参考文献：

[1] Farabet C, Poulet C, Han J, *et al.* CNP: an FPGA-based processor for convolutional networks [C]// Proc of International Conference on Filed Programmable Logic and Application. Piscataway, NJ: IEEE Press, 2009: 32-37.

[2] Devlin J, Zbib R, Huang Zhongqiang, *et al.* Fast and robust neural network joint models for statistical machine translation [C]// Proc of ACL Conference. 2014: 1370-1380.

[3] Geoffrey H, Li Deng, Dong Yu, *et al.* Deep neural networks for acoustic modeling in speech recognition [J]. IEEE Signal Processing Magazine, 2012, 29 (6): 82-97.

[4] Garcia C, Delakis M. Convolutional face finder: a neural architecture for fast and robust face detection [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2004, 26 (11): 1408-1423.

[5] Courbariaux M, Bengio Y, David J P. BinaryConnect: training deep neural networks with binary weights during propagations [C]. Proc of the 28th International Conference on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2015: 3123-3131.

[6] Courbariaux M, Bengio Y. Binarized neural networks: training deep neural networks with weights and activations constrained to+1 or-1 [J]. arXiv: 1602. 02830v1, 2016.

[7] Hubara I, Courbariaux M, Bengio Y, *et al.* Quantized neural networks: training neural networks with low precision weights and activeations [J]. arXiv preprint arXiv: 1609. 07061, 2016.

[8] Cong J, Bin Liu, Neuendorffer S, *et al.* High-level synthesis for fpgas: from prototyping to deployment [J]. IEEE Trans on Computer Aided Design of Integrated Circuits and Systems, 2011, 30 (4): 473-491.

[9] Bottleson J, Sungye K, Andrews J, *et al.* ClCaffe: OpenCL accelerated caffe for convolutional neural networks [C]// Proc of IEEE International Parallel and Distributed Processing Symposium Workshops. 2016: 50-57.

[10] Andrew L, Gray S. Fast algorithms for convolutional neural networks [J]. arXiv e-print, arXiv: 1509. 09308.

[11] Jong H K, Mudassar B, Taesik N, *et al.* Design of an energyefficient accelerator for training of convolutional neural networks using frequency-domain computation [C]// Proc of the 54th Annual Conference on Design Automation. 2017.

[12] Chakradhar S, Sankaradas M, Jakkula V, *et al.* A dynamically configurable coprocessor for convolutional neural networks [J]. ACM SIGARCH Computer Architecture News, 2010, 38 (3): 247-257.

[13] Li Huimin, Fan Xitian, Jiao Li, *et al.* A high performance FPGA-based accelerator for large-scale convolutional neural networks. [C]// Proc of the 26th International Conference on Field Programmable Logic and Applications. 2016: 1-9.

[14] Wei Xuechao, Yu C H, Zhang Peng, *et al.* Auto-mated systolic array architecture synthesis for high throughput CNN inference on FPGAs [C]// Proc of Annual Conference on Design Automation. 2017: 1-6.

[15] Derrien S, Rajopadhye S. Loop tiling for reconfigurable accelerators [C]// Proc of International Conference on Field Programmable Logic and Applications. 2001: 398-408.

[16] Williams S, Waterman A, Patterson D. Roof line: an insightful visual performance model for multicore architectures [J]. Communicat ions of the ACM, 2009, 52 (4): 65-76.

[17] Han S, Pool J, Tran J, *et al.* Learning both weights and connections for efficient neural network [C]// Advances in Neural Information Processing Systems. 2015: 1135-1143.

[18] Sironi A, Tekin B, Rigamonti R, *et al.* Learning separable filters [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2015, 37 (1): 94-106.

[19] Farabet C, Poulet C, LeCun Y, *et al.* CNP: an FPGA-based processor for convolutional networks [C]// Proc of International Conference on Field Programmable Logic and Applications. 2009: 1689-1699.

[20] Williamson D. Dynamically scaled fixed point arithmetic [C]// Proc of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing Conference. 1991: 315-318.

[21] Rastegari M, Ordonez V, Redmon J, *et al.* XNOR-net: imagenet classification using binary convolutional neural networks [C]// Proc of European Conference on Computer Vision. 2016: 525-542.

[22] Umuroglu Y, Nicholas J F, Gambardella G, *et al.* FINN: a framework for fast scalable binarized neural network inference [C]// Proc of ACM/SIGDA International Symposium on Fi eld Programmable Gate Arrays. 2017: 65-74.

[23] Hubara I, Courbariaux M, Bengio Y. Binarized neural networks [C]// Advances in Neural Information Processing System. 2016: 4107-4115.

[24] Li Fengfu, Zhang Bo, Liu Bin. Ternary weight networks [J]. arXiv e-print, arXiv: 1605. 04711, 2016.

[25] Ren Ao, Li, Zhe, Ding Caiwen, *et al.* SC-DCNN: highly-scalable deep convolutional neural network using stochastic computing [C]// Proc of International Conference on Architectural Support for Program ming Languages and Operating Systems. 2017: 405-418.

[26] Ioffeand S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift [J]. arXiv eprint, arXiv: 1502. 03167, 2015.

[27] Krizhevsky, Hinton G. Learning multiple layers of features from tiny images [D]. [S. l.] : University of Toronto, 2009.

[28] Suda N, Chandra V, Dasika G, *et al*. Throughput-optimal OpenCL based FPGA accelerator for large-scale convolutional neural networks [C]// Proc of ACM/SIGDA ISFPGA. 2016: 16-25.

[29] QiuJiantao, Wang Jie, Yao Song, *et al.* Going deeper with embedded FPGA platform for convolutional neural network [C]// Proc of ACM/SIGDA International Symposium on Field-Programmable Gate Array. 2016: 26-35.

[30] Venieris S I, Bouganis C S. fpgaConvNet: a framework for mapping convolutional neural networks on FPGAs [C]// Proc of IEEE FCCM. 2016: 40-47.